Similarity of Neural Architectures using Adversarial Attack Transferability

Jaehui Hwang[1,2,†] Dongyoon Han[3] Byeongho Heo[3] Song Park[3]
Sanghyuk Chun[3,*] Jong-Seok Lee[1,2,*]

[1] School of Integrated Technology, Yonsei University
[2]BK21 Graduate Program in Intelligent Semiconductor Technology, Yonsei University
[3] NAVER AI Lab

[†] Works done during an internship at NAVER AI Lab. [*] Corresponding authors

Abstract. In recent years, many deep neural architectures have been developed for image classification. Whether they are similar or dissimilar and what factors contribute to their (dis)similarities remains curious. To address this question, we aim to design a quantitative and scalable similarity measure between neural architectures. We propose Similarity by Attack Transferability (SAT) from the observation that adversarial attack transferability contains information related to input gradients and decision boundaries widely used to understand model behaviors. We conduct a large-scale analysis on 69 state-of-the-art ImageNet classifiers using our SAT to answer the question. In addition, we provide interesting insights into ML applications using multiple models, such as model ensemble and knowledge distillation. Our results show that using diverse neural architectures with distinct components can benefit such scenarios.

Keywords: Architecture Similarity · Adversarial Attack Transferability

1 Introduction

The advances in deep neural networks (DNN) architecture design have taken a key role in their success by making the learning process easier (*e.g.*, normalization [2, 49, 106] or skip connection [39]), enforcing human inductive bias [56], or increasing model capability with the self-attention mechanism [98]. With different architectural components containing architectural design principles and elements, a number of different neural architectures have been proposed. They have different accuracies, but several researches have pointed out that their predictions are not significantly different [33, 66, 67].

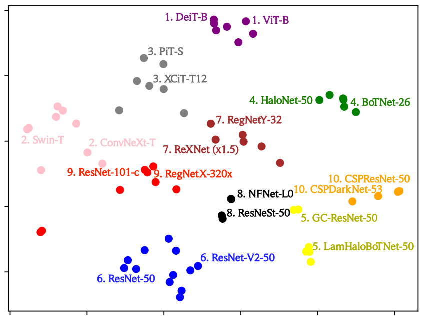


Fig. 1: t-SNE plot showing 10 clusters of 69 neural networks using our similarity function, SAT.

By this, *can we say that recently developed DNN models with different architectural components are similar or the same?* The answer is *no*. It is because a model prediction is not the only characteristic to compare their similarities. Existing studies have found differences by focusing on different features, such as layer-by-layer network component [55, 75], a high-level understanding by visualization of loss surface [26], input gradient [82, 84], and decision boundary [83]. Researchers could understand the similarity between models through these trials; however, the similarity comparison methods from previous studies are insufficient for facilitating comprehensive studies because they do not satisfy two criteria that practical metrics should meet: (1) providing a quantitative similarity score and (2) being compatible with different base architectures (*e.g.*, CNN and Transformer). Recently, Tramèr et al. [95] and Somepalli et al. [83] suggested a quantitative similarity metric based on measuring differences in decision boundaries. However, these methods have limitations due to the non-tractable decision boundaries and limited computations as shown in Sec. 3.

We propose a quantitative similarity that is scalable and easily applicable to diverse architectures, named Similarity by Attack Transferability (SAT). We focus on adversarial attack transferability (AT), which indicates how generated adversarial perturbation is transferable between two different architectures. It is widely studied that the vulnerability of DNNs depends on their own architectural property or how models capture the features from inputs, such as the usage of self-attention [31], the stem layer [47], and the dependency on high or low-frequency components of input [3, 54]. Thus, if two different models are similar, the AT between the models is high because they share similar vulnerability [77]. Furthermore, AT can be a reliable approximation for comparing the input gradients [65], decision boundary [53], and loss landscape [25]. All of them are widely-used frameworks to understand model behavior and differences between models and used to measure the similarity of models in previous works [5, 17, 26, 59, 82, 83, 84, 87, 95, 104]; namely, SAT can capture various model properties.

We quantitatively measure pairwise SATs of 69 different ImageNet-trained neural architectures from [105]. We analyze what components among 13 architectural components (*e.g.*, normalization, activation, . . .) that consist of neural architectures largely affect model diversity. Furthermore, we observe relationships between SAT and practical applications, such as ensemble and distillation.

2 Related Work

Similarity between DNNs has been actively explored recently. Several studies focused on comparing intermediate features to understand the behavior of DNNs. Raghu et al. [75] observed the difference between layers, training methods, and architectures (*e.g.*, CNN and ViT) based on **layer-by-layer comparison** [55]. Some studies have focused on **loss landscapes** by visualizing the loss of models on the parameter space [26, 59, 73]. Although these methods show a visual inspection, they cannot support quantitative measurements. On the other hand, our goal is to support a quantitative similarity by SAT.

Another line of research has been focused on **prediction-based statistics**, *e.g.*, comparing wrong and correct predictions [32, 33, 57, 79]. However, as recent complex DNNs are getting almost perfect, just focusing on prediction values can be misleading; Meding et al. [67] observed that recent DNNs show highly similar predictions. In this case, prediction-based methods will be no more informative. Meanwhile, our SAT can provide meaningful findings for 69 recent NNs.

Input gradient is another popular framework to understand model behavior by observing how a model will change predictions by local pixel changes [5, 81, 82, 84, 87]. If two models are similar, their input gradients will also be similar. These methods are computationally efficient, and no additional training is required; they can provide a visual understanding of the given input. However, input gradients are inherently noisy; thus, these methods will need additional pre-processing, such as smoothing, for a stable computation [17]. Also, these methods usually measure how the input gradient matches the actual foreground, *i.e.*, we need ground-truth foreground masks for measuring such scores. On the contrary, SAT needs no additional pre-processing and mask annotations.

Comparing the decision boundaries will provide a high-level understanding of how models behave differently for input changes and how models extract features from complicated data dimensions. Recent works [95, 104] suggested measuring similarity by comparing distances between predictions and decision boundaries. Meanwhile, Somepalli et al. [83] analyzed models by comparing their decision boundaries on the on-manifold plane constructed by three random images. However, these approaches suffer from inaccurate approximation, non-tractable decision boundaries, and finite pairs of inputs and predictions.

Finally, different behaviors of CNNs and Transformers have been studied in specific tasks, such as robustness [4, 69], layer-by-layer comparison [73, 75] or decision-making process [50]. Our work aims to quantify the similarity between general NNs, not only focusing on limited groups of architecture.

3 Similarity by Attack Transferability (SAT)

Here, we propose a quantitative similarity between two architectures using adversarial attack transferability, which indicates whether an adversarial sample from a model can fool another model. The concept of adversarial attack has effectively pointed out the vulnerabilities of DNNs by input gradient [34, 65, 88].

Interestingly, these vulnerabilities have been observed to be intricately linked to architectural properties. For example, Fu et al. [31] demonstrated the effect of the attention modules in architecture on attack success rate. Hwang et al. [47] analyzed that the stem layer structure causes models to have different adversarial vulnerable points in the input space, *e.g.*, video models periodically have vulnerable frames, such as every four frames. Namely, an adversarial sample to a model highly depends on the inherent architectural property of the model.

Another perspective emphasized the dissimilarities in dependencies on high-frequency and low-frequency components between CNN-based and transformer-based models, showing different vulnerabilities to different adversarial attacks

[3, 54]. Different architectural choices behave as different frequency filters (*e.g.*, the self-attention works as a low-pass filter, while the convolution works as a high-pass filter) [73]; thus, we can expect that the different architectural component choices will affect the model vulnerability, *e.g.*, vulnerability to high-frequency perturbations. If we can measure how the adversarial vulnerabilities of the models are different, we also can measure how the networks are dissimilar.

To measure how model vulnerabilities differ, we employ **adversarial attack transferability** (AT), where it indicates whether an adversarial sample from a model can fool another model. If two models are more similar, their AT gets higher [25, 60, 77]. On the other hand, because the adversarial attack targets vulnerable points varying by architectural components of DNNs [31, 46, 47, 54], if two different models are dissimilar, the AT between them gets lower. Furthermore, attack transferability can be a good approximation for measuring the differences in input gradients [65], decision boundaries [53], and loss landscape [25], where they are widely used techniques for understanding model behavior and similarity between models as discussed in the related work section. While previous approaches are limited to non-quantitative analysis, inherent noisy property, and computational costs, adversarial transferability can provide quantitative measures with low variances and low computational costs.

We propose a new similarity function that utilizes attack transferability, named **Similarity by Attack Transferability (SAT)**, providing a reliable, easy-to-conduct, and scalable method for measuring the similarity between neural architectures. Formally, we generate adversarial samples x_A and x_B of model A and B for the given input x. Then, we measure the accuracy of model A using the adversarial sample for model B (called $\mathrm{acc}_{B\to A}$). If A and B are the same, then $\mathrm{acc}_{B\to A}$ will be zero if the adversary can fool model B perfectly. On the other hand, if the input gradients of A and B differ significantly, then the performance drop will be neglectable because the adversarial sample is almost similar to the original image (*i.e.*, $\|x - x_B\| \le \varepsilon$). Let X_{AB} be the set of inputs where both A and B predict correctly, y be the ground truth label, and $\mathbb{I}(\cdot)$ be the indicator function. We measure SAT between two different models by:

$$\mathrm{SAT}(A,B) = \log\Big[\max\Big\{\varepsilon_s, 100\times\frac{1}{2|X_{AB}|}\sum_{x\in X_{AB}}\{\mathbb{I}(A(x_B)\neq y) + \mathbb{I}(B(x_A)\neq y)\}\Big\}\Big], \tag{1}$$

where ε_s is a small scalar value. If $A = B$ and we have an oracle adversary, then $\mathrm{SAT}(A, A) = \log 100$. In practice, a strong adversary (*e.g.*, PGD [65] or AutoAttack [22]) can easily achieve a nearly-zero accuracy if a model is not trained by an adversarial attack-aware strategy [21, 65]. Meanwhile, if the adversarial attacks on A are not transferable to B and vice versa, then $\mathrm{SAT}(A, B) = \log \varepsilon_s$.

Ideally, we aim to define a similarity d between two models with the following properties: (1) $n = \arg\min_m d(n, m)$, (2) $d(n, m) = d(m, n)$ and (3) $d(n, m) > d(n, n)$ if $n \neq m$. If the adversary is perfect, then $\mathrm{acc}_{A\to A}$ will be zero, and it will be the minimum because accuracy is non-negative. "$\mathrm{acc}_{A\to B} + \mathrm{acc}_{B\to A}$" is symmetric thereby SAT is symmetric. Finally, SAT satisfies $d(n, m) \ge d(n, n)$ if $n \neq m$ where it is a weaker condition than (3).

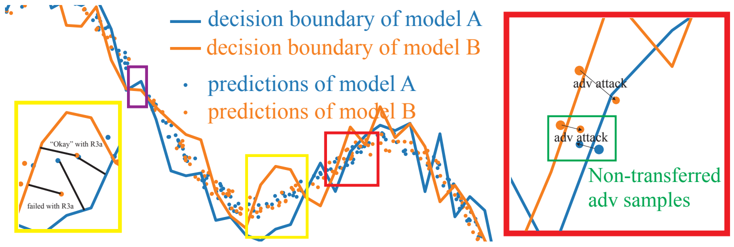


Fig. 2: How SAT works? Conceptual figure to understand SAT by the lens of the decision boundary. Each line denotes the decision boundary of a binary classification model, and each dot denotes individual prediction for given inputs.

Comparison with other methods. Here, we compare SAT with prediction-based measurements [32, 33, 57, 79] and similarity measurements by comparing decision boundaries (Tramèr et al. [95] and Somepalli et al. [83]). We first define two binary classifiers f and g and their predicted values $f_p(x)$ and $g_p(x)$ for input x (See Fig. 2). f classifies x as positive if $f_p(x) > f_d(x)$ where $f_d(x)$ is a decision boundary of f. We aim to measure the difference between decision boundaries, namely $\int_x |f_d(x) - g_d(x)| dx$ to measure differences between models. However, DNNs have a non-tractable decision boundary function, thus, f_d and g_d are not tractable. Furthermore, the space of x is too large to compute explicitly. Instead, we may assume that we only have finite and sparingly sampled x.

In this scenario, we can choose three strategies. First, we can count the number of samples whose predicted labels are different for given x, which is *prediction-based measurements* or Somepalli et al. [83]. As we assumed sparsity of x, this approach cannot measure the area of uncovered x domain, hence, its approximation will be incorrect (purple box in Fig. 2) or needs too many perturbations to search uncovered x. In Appendix A.1, we empirically show that Somepalli et al. [83] suffers from the high variance even with a large number of samples while SAT shows a low variance with a small number of samples.

Second, we can measure the minimum distance between $f_p(x)$ and f_d as Tramèr et al. [95]. This only measures the distance to its closest decision boundary without considering the other model. As shown in the yellow box of Fig. 2, if two predictions are similar at x, it would compute an approximation of $|f_d(x) - g_d(x)|$ for x. However, if two predictions are different, it will compute a wrong approximation. Moreover, in practice, searching ϵ is unstable and expensive.

Lastly, we can count the number of non-transferred adversarial samples (red box in Fig. 2), which is *our method, SAT*. If we have an oracle attack method that exactly moves the point right beyond the decision boundary, our SAT will measure the ℓ_0 approximation of $\min(|f_d(x) - g_d(x)|, \epsilon)$ for given x. Namely, SAT can measure whether two decision boundaries are different by more than ϵ for each x. If we assume that the difference between decision boundaries is not significantly large and ϵ is properly chosen, SAT will compute an approximated decision boundary difference. We also compare SAT and other methods from the viewpoint of stability and practical usability in Sec. 5.1 and Appendix A.

Discussions. In practice, we do not have an oracle attack method. Instead, we employ the PGD attack [65] as the adversarial attack method. In Appendix B.1, we investigate the robustness of SAT to the choice of the attack methods. In summary, SAT measured by PGD shows a high correlation with SAT measured by various attacks, *e.g.*, AutoAttack [22], attacks designed for enhancing attack transferability, such as MIFGSM [27] and VMIFGSM [103], low-frequency targeted attacks, such as low-frequency PGD [36], method-specific attacks, such as PatchPool [31], or generative model-based attacks, such as BIA [116].

Also, SAT assumes an optimal attack with proper ϵ. However, this assumption can be broken under the adversarial training setting when we use a practical attacker. Also, as shown by Tsipras et al. [97] and Ilyas et al. [48], adversarial training will lead to a different decision boundary from the original model. In Appendix B.2, we empirically investigate the effect of adversarial training to SAT. We observe that different adversarial training methods make as a difference as different training techniques, which we will discuss in Sec. 4.2.

Analyzing 69 models. Now, we analyze 69 recent ImageNet classifiers using SAT by focusing on two questions. (1) Which network component contributes to the diversity between models? (2) Why do we need to develop various neural architectures? The full list of the architectures can be found in Appendix C. We use the PGD attack [65] for the adversary. We set the iteration to 50, the learning rate to 0.1, and ε to 8/255. As we discussed earlier, we show that SAT is robust to the choice of the adversarial attack method. We select 69 neural architectures trained on ImageNet [78] from the PyTorch Image Models library [105]. To reduce the unexpected effect of a significant accuracy gap, the chosen model candidates are limited to the models whose top-1 accuracy is between 79% and 83%. We also ignore the models with unusual training techniques, such as training on extra training datasets, using a small or large input resolution (*e.g.*, less than 200 or larger than 300), or knowledge distillation. When A and B take different input resolutions, then we resize the attacked image from the source network for the target network. We also sub-sample 10% ImageNet validation images (*i.e.*, 5,000 images) to measure the similarity. This strategy makes our similarity score more computationally efficient.

4 Model Analysis by Network Similarity

4.1 Which Architectural Component Causes the Difference?

Settings. We list 13 key architecture components: normalization (*e.g.*, BN [49] and LN [2]), activations (*e.g.*, ReLU [56] and GeLU [76]), the existence of depthwise convolution, or stem layer (*e.g.*, 7×7 conv, 3×3 conv, or 16×16 conv with stride 16 – a.k.a. *"patchify"* stem [64]). The list of the entire components is shown in the Appendix. We then convert each architecture as a feature vector based on the listed sub-modules. For example, we convert ResNet as $f_{\text{ResNet}} = [\text{Base arch} = \text{CNN}, \text{Norm} = \text{BN}, \text{Activation} = \text{ReLU}, \ldots]$. The full list of components of 69 architectures can be found in Appendix C.

Table 1: Clusters by SAT. All the architectures here are denoted by the aliases defined in their respective papers. We show the top-5 keywords for each cluster based on TF-IDF. InRes, SA, and CWA denote input resolution, self-attention, and channel-wise attention, respectively. The customized model details are described in the footnote†.

No.	Top-5 Keywords	Architecture
1	Stem layer: 16×16 conv w/ s16, No Hierarchical, GeLU, LN, Final GAP	`ConViT-B` [29], `CrossViT-B` [12],`DeiT-B` [93], `DeiT-S` [93], `ViT-S (patch size 16)` [28],`ResMLP-S24` [94], `gMLP-S` [62]
2	Stem layer: 4×4 conv w/ s4, LN, GeLU, Transformer, No pooling at stem	`Twins-PCPVT-B` [19], `Twins-SVT-S` [19], `CoaT-Lite Small` [24], `NesT-T` [118], `Swin-T` [63], `S3 (Swin-T)` [13], `ConvNeXt-T` [64],`ResMLP-B24` [94]
3	Transformer, Final GAP, GeLU, Pooling at stem, InRes: 224	`XCiT-M24` [1], `XCiT-T12` [1], `HaloRegNetZ-B`**, `TNT-S` [38], `Visformer-S` [16], `PiT-S` [43], `PiT-B` [43]
4	Stem layer: stack of 3×3 conv, 2D SA, InRes: 256, Pooling at stem, SiLU	`HaloNet-50` [99], `LambdaResNet-50` [6], `BoTNeT-26` [85], `GC-ResNeXt-50` [11], `ECAHaloNeXt-50`**, `ECA-BoTNeXt-26`**
5	Stem layer: stack of 3×3 convs, InRes: 256, 2D SA, CWA: middle of blocks, CNN	`LamHaloBoTNet-50`**, `SE-BoTNet-33`**, `SE-HaloNet-33`**, `Halo2BoTNet-50`**, `GC-ResNet-50` [11], `ECA-Net-33` [102]
6	Stem layer: 7×7 conv w/ s2, ReLU, Pooling at stem, CNN, BN	`ResNet-50` [39], `ResNet-101` [39], `ResNeXt-50` [107], `Wide ResNet-50` [112], `SE-ResNet-50` [45], `SE-ResNeXt-50` [45], `ResNet-V2-50` [40], `ResNet-V2-101` [40], `ResNet-50 (GN)` [106], `ResNet-50 (BlurPool)` [117], `DPN-107` [15], `Xception-65` [18]
7	NAS, Stem layer: 3×3 conv w/ s2 CWA: middle of blocks, CWA, DW Conv	`EfficientNet-B2` [90], `FBNetV3-G` [23], `ReXNet` (×1.5) [37], `RegNetY-32` [74], `MixNet-XL` [91], `NF-RegNet-B1` [9]
8	Input resolution: 224, Stem layer: stack of 3×3 convs, Group Conv, Final GAP, 2D SA	`NFNet-L0`**, `ECA-NFNet-L0`**, `PoolFormer-M48` [110], `ResNeSt-50` [114], `ResNet-V2-50-D-EVOS`**, `ConvMixer-1536/20` [96]
9	ReLU, Input resolution: 224, DW Conv, BN, 2D self-attention	`ViT-B (patch size 32)` [28], `R26+ViT-S` [86], `DLA-X-102` [109], `eSE-VoVNet-39` [58], `ResNet-101-C` [42], `RegNetX-320` [74], `HRNet-W32` [101]
10	ReLU + Leaky ReLU, InRes: 256, Stem layer: 7×7 conv, CNN, Pooling at stem	`CSPResNet-50` [100], `CSPResNeXt-50` [100], `CSPDarkNet-53` [7], `NF-ResNet-50` [9]

Feature important analysis. Now, we measure the feature importance by fitting a gradient boosting regressor [30] on the feature difference ($e.g.$, $f_{\text{ResNet-50}} - f_{\text{DeiT-base}}$) measured by Hamming distance and the corresponding similarity. The details of the regressor are described in Appendix. We use the permutation importance [8] that indicates how the trained regression model changes the prediction according to randomly changing each feature. The feature importance of each architectural component is shown in Fig. 3. We first observe that the choice of base architecture ($e.g.$, CNN [56], Transformer [98], and MLP-Mixer [92]) contributes to the similarity most significantly. Fig. 3 also shows that the design choice of the input layer ($i.e.$, stem layer design choice or input resolution) affects the similarity as much as the choice of basic components such as normalization layers, activation functions, and the existence of attention layers. On the other hand, we observe that the modified efficiency-aware convolution operations, such as depth-wise convolution [18], are ineffective for diversity.

Clustering analysis. We additionally provide a clustering analysis based on the architectural similarities. We construct a pairwise similarity graph with adjacency matrix A between all 69 architectures where its vertex denotes an architecture, and its edge denotes the similarity between two networks. We perform

** Customized models by [105]: `HaloRegNetZ` = `HaloNet` + `RegNetZ`; `ECA-BoTNeXt` = `ECA-Net` + `HaloNet` + `ResNeXt`; `ECA-BoTNeXt` = `ECA-Net` + `BoTNet` + `ResNeXt`; `LamHaloBoTNet` = `LambdaNet` + `HaloNet` + `BoTNet`; `SE-BoTNet` = `SENet` + `BoTNet`; `SE-HaloNet` = `SENet` + `HaloNet`; `Halo2BoTNet` = `HaloNet` + `BoTNet`; `NFNet-L0` = an efficient variant of `NFNet-F0` [10]; `ECA-NFNet-L0` = `ECA-Net` + `NFNet-L0`; `ResNet-V2-D-EVOS` = `ResNet-V2` + `EvoNorms` [61].

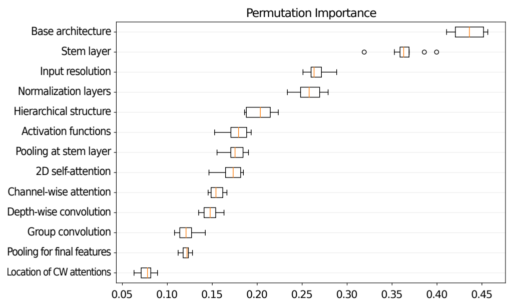


Fig. 3: Importance of architectural components to network similarity. 13 components are sorted by the contribution to the similarities. The larger feature importance means the component contributes more to the network similarity.

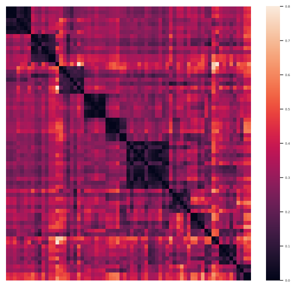


Fig. 4: Pairwise distances of spectral features. Rows and columns are sorted by the clustering index. More details are described in Appendix.

the spectral clustering [70] on A where the number of clusters K is set to 10: We compute the Laplacian matrix of A, $L = D - A$ where D is the diagonal matrix and its i-th component is $\sum_j A_{ij}$. Then, we perform K-means clustering on the K-largest eigenvectors of L. The pairwise distances of spectral features (*i.e.*, 10-largest eigenvectors of L) of 69 neural architectures are shown in Fig. 4. The rows and columns of Fig. 4 are sorted by the clustering index (Tab. 1). More details with model names are described in Appendix D.2. We can see the block-diagonal patterns, *i.e.*, in-clusters similarities are more significant than between-clusters similarities. More details are in Appendix D.3.

Tab. 1 shows the clustering results on 69 networks and the top-5 keywords for each cluster based on term frequency-inverse document frequency (TF-IDF) analysis. Specifically, we treat each model feature as a word and compute TF and IDF by treating each architecture as a document. Then we compute the average TF-IDF for each cluster and report top-5 keywords. Similar to Fig. 3, the base architecture (*e.g.*, CNN in Cluster 5, 6, 10 and Transformer in Cluster 2, 3) and the design choice for the stem layer (*e.g.*, Cluster 1, 2, 4, 5, 6, 7, 8, 10) repeatedly appear at the top keywords. Especially, we can observe that the differences in base architecture significantly cause the diversity in model similarities, *e.g.*, non-hierarchical Transformers (Cluster 1), hierarchical networks with the patchification stem (Cluster 2), hierarchical Transformers (Cluster 3), CNNs with 2D self-attention (Cluster 4, 5), ResNet-based architectures (Cluster 6), and NAS-based architectures (Cluster 7).

4.2 The Relationship between Training Strategy and SAT

The architectural difference is not the only cause of the model diversity. We compare the impact by different architecture choices (*e.g.*, ResNet and ViT) and by different training strategies while fixing the model architecture, as follows: **Different initializations** can affect the model training by the nature of the stochas-

Table 2: SAT within the same architecture. We compare the average similarity within the same architecture but trained with different procedures, "All" denotes the average similarity of 69 architectures.

Architecture	`ResNet-50`	`ViT-S`
Init	4.23	4.21
Hparam	4.05	4.22
Tr. Reg.	3.27	3.44
All	2.73	

Table 3: Ensemble performance with diverse architectures. We report the error reduction rate by varying the number of ensembled models and the diversity of the ensemble models (related to Fig. 7a). "rand" indicates the random choice of models.

	less diverse ← # of clusters → more diverse					
# of models	1	2	3	4	5	rand
2	7.13	**7.84**				7.78
3	10.17	10.84	**11.20**			11.11
4	11.70	12.45	12.80	**13.00**		12.90
5	12.58	13.41	13.79	13.99	**14.11**	14.01

ticity of the training procedure. For example, Somepalli et al. [83] showed that the decision boundary of each architecture could vary by different initializations. We also consider **different optimization hyper-parameters** (*e.g.*, learning rate, weight decay). Finally, we study the effect of **different training regimes** (*e.g.*, augmentations, type of supervision). For example, the choice of data augmentation [111, 113] or label smoothing [89] can theoretically or empirically affect adversarial robustness [20, 72, 80, 115]. We also investigate the effect of supervision, such as self-supervision [14, 35, 41] or semi-weakly supervised learning [108]. Note that the training strategies inevitably contain the former ones. For example, when we train models with different training regimes, models have different initialization seeds and different optimization hyper-parameters. Comparing 69 different architectures also contains the effect of different initialization and optimization hyper-parameters and parts of different training regimes. This is necessary for achieving high classification performance.

Tab. 2 shows the comparison of similarity scores between the same architecture but different learning methods (a smaller similarity means more diversity). We report two architectures, ResNet-50 and ViT-S, and their training settings are in Appendix E. We also show the average SAT between all 69 architectures. In the table, we first observe that using different random initialization or different optimization hyper-parameters shows high correlations with each other (almost ≥ 4.2) while the average similarity score between various neural architectures is 2.73. In other words, the difference in initializations or optimization hyper-parameters does not significantly contribute to the model diversity.

Second, we observe that using different learning techniques remarkably affects SAT (3.27 for ResNet and 3.44 for ViT), but is not as significant as the architectural difference (2.73). Furthermore, the change of SAT caused by different initializations or hyper-parameters is less marked than the change caused by different architecture (Fig. 5). These observations provide two insights. First, the diversity resulting from various training strategies is not significant enough compared to the diversity of architecture. Second, designing new architecture is more efficient in achieving diverse models rather than re-training the same one.

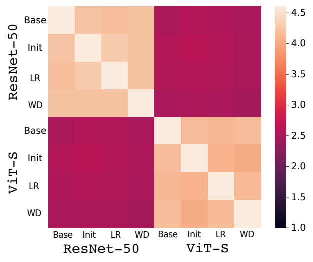


Fig. 5: Pairwise distance of spectral features by different optimizations. `Init`, `LR`, and `WD` are randomly chosen from models trained with different settings of initialization, learning rate, and weight decay in Tab. 2.

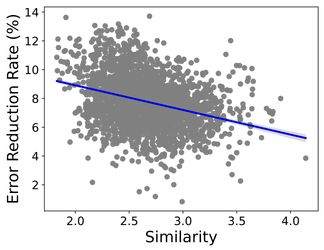


Fig. 6: Correlation between pairwise SAT and ensemble performance. The trend line and its 90% confidence interval are shown.

5 SAT Applications with Multiple Models

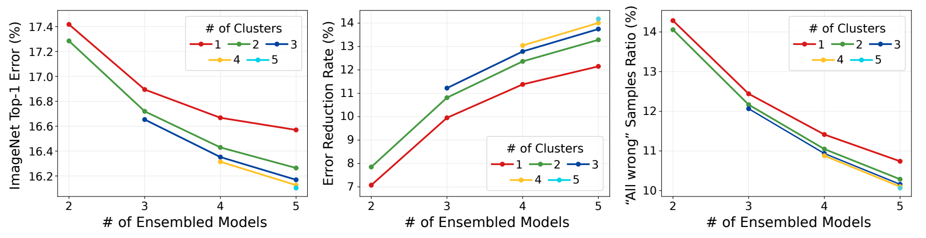
Here, we analyze how SAT is related to downstream tasks involving more than one model. First, we show that using more diverse models will lead to better ensemble performance. Second, we study the relationship between knowledge distillation and SAT. Furthermore, we can suggest a similarity-based guideline for choosing a teacher model when distilling to a specific architecture. Through these observations, we can provide insights into the necessity of diverse models.

5.1 Model Diversity and Ensemble

Settings. The model ensemble is a practical technique for achieving high performance. However, only few works have studied the relationship between ensemble performance and model similarity, particularly for large-scale complex models. Previous studies are mainly conducted on tiny datasets and linear models [57]. We investigate the change of ensemble performance by the change of similarity based on the unweighted average method [52] (*i.e.*, averaging the logit values of the ensembled models). Because the ensemble performance is sensitive to the original model performances, we define Error Reduction Rate (ERR) as $1 - \frac{\text{Err}_{\text{ens}}(M)}{\frac{1}{|M|}\sum_{m \in M} \text{Err}(m)}$, where M is the set of the ensembled models, $\text{ERR}(m)$ denotes the top-1 ImageNet validation error of model m, and $\text{Err}_{\text{ens}}(\cdot)$ denotes the top-1 error of the model ensemble results.

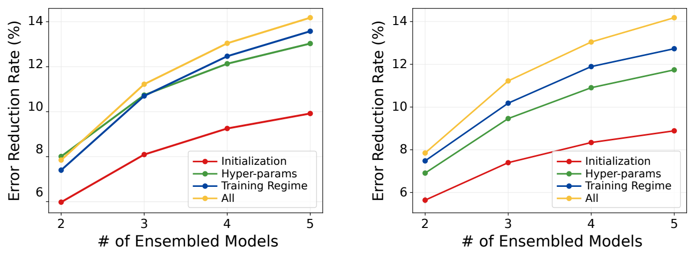
Results. We first measure the 2-ensemble performances among the 69 architectures (*i.e.*, the number of ensembles is $\binom{69}{2} = 2346$). We plot the relationship between SAT and ERR in Fig. 6. We observe that there exists a strong negative correlation between the model similarity and the ensemble performance (Pearson correlation coefficient -0.32 with p-value ≈ 0 and Spearman correlation -0.32 with p-value ≈ 0, *i.e.*, *more diversity leads to better ensemble performance.*

We also conduct N-ensemble experiments with $N \geq 2$ based on our clustering results in Tab. 1. We evaluate the average ERR of the ensemble of models from



(a) Top-1 Error **(b)** Error reduction rate **(c)** "All wrong" samples ratio

Fig. 7: Model diversity and ensemble performance. We report ensemble performances by varying the number of ensembled models (N) and the diversity of the models. The diversity is controlled by choosing the models from k different clusters.



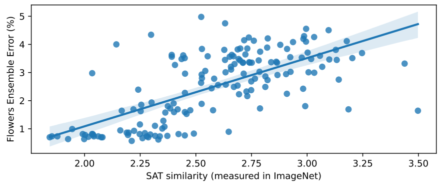
(a) `ResNet-50` **(b)** `ViT-S`

Fig. 8: Diversity by training techniques and ensemble. We report the the same metrics as Fig. 7 for various `ResNet-50` and `ViT-S` models in Tab. 2.

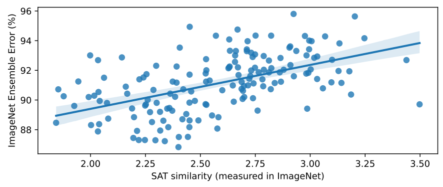
k clusters, *i.e.*, if $N = 5$ and $k = 3$, the ensembled models are only sampled from the selected 3 clusters while ignoring the other 7 clusters. We investigate the effect of model diversity and ensemble performance by examining $k = 1 \ldots N$ (*i.e.*, larger k denotes more diverse ensembled models). We report the result with ImageNet top-1 error and ERR in Fig. 7a and Fig. 7b.

In all metrics, we observe that the ensemble of more diverse models shows better performance. Interestingly, Fig. 7b shows that when the number of clusters for the model selection (k) is decreased, the ensemble performance by the number of ensembled models (N) quickly reaches saturation. Tab. 3 shows that the ensemble performances by choosing the most diverse models via SAT always outperform the random ensemble. Similarly, Fig. 7c shows that the number of wrong samples by all models is decreased by selecting more diverse models.

Training Strategy vs. Architecture in the ensemble scenario? Remark that Tab. 2 showed that the different training strategies are not as effective as different architectures for diversity. To examine this on the ensemble scenario, we report the ensemble results of different training strategies, *i.e.*, the same `ResNet-50` and `ViT-S` in Tab. 2. For comparison with different architectures, we also report the ensemble of different architectures where all ensembled models are from different clusters (*i.e.*, N=k in Fig. 7). Fig. 8 shows that although using diverse



(a) SAT vs. Flowers ensemble



(b) SAT vs. ImageNet ensemble

Fig. 9: Cross-dataset SAT results. SAT measured on ImageNet also has a positive correlation with ensemble performances on Flowers-102 [71].

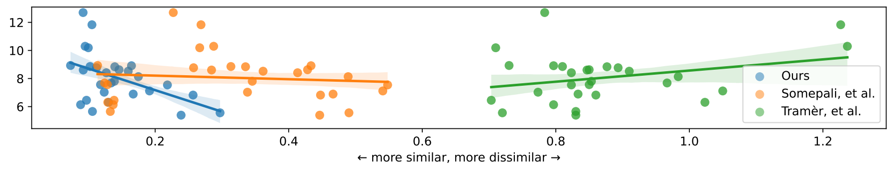


Fig. 10: Empircial comparison. Relationship between the model similarity, including SAT, Somepalli et al. [83] and Tramèr et al. [95], and 2-ensemble performance.

training regimes (blue lines) improves ensemble performance compared to other techniques (red and green lines), the improvements by using different architectures (yellow lines) are more significant than the improvements by using different training regimes (blue lines) with large gaps.

Generalizability to other datasets. We examine whether more diverse architectures in ImageNet SAT also lead to better ensemble performances on the other datasets. We fine-tuned all 69 architectures to the Flowers-102 dataset [71], and filter out low performing models ($<$ 95% top-1 accuracy). After the filtering, we have 16 fine-tuned models. Using the fine-tuned models, we plot the relationship between the Flowers-102 ensemble performance and SAT score measured in ImageNet. Fig. 9 shows that SAT also highly correlates with Flowers ensemble performances, despite that SAT is measured on ImageNet. This experimental result supports that SAT similarity can be applied in a cross-domain manner.

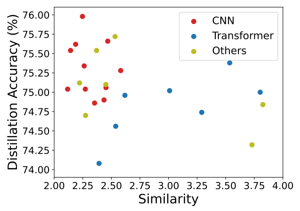
Comparison of different similarity functions in the ensemble scenario. Finally, we compare the impact of the choice of the similarity function and the ensemble performance when following our setting. We compare SAT with Somepalli et al. [83] and Tramèr et al. [95] on the 2-ensemble scenario with 8 out of 69 models due to the stability issue of Tramèr et al. [95]. Fig. 10 shows the relationship between various similarity functions and the 2-ensemble performance. We observe that SAT only shows a strong positive correlation (blue line), while the others show an almost random or slightly negative correlation. Finally, in Appendix A.2, we compare SAT with Somepalli et al. [83] and a naive architecture-based clustering using our features under the same setting of Fig. 7 and 8. Similarly, SAT shows the best ensemble performance against the comparison methods.

5.2 Model Diversity and Knowledge Distillation

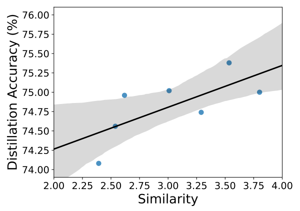
Knowledge distillation (KD) [44] is a training method for transferring rich knowledge of a well-trained teacher network. Intuitively, KD performance affects a lot by choice of the teacher network; however, the relationship between similarity and KD performance has not yet been explored enough, especially for ViT. This subsection investigates how the similarity between teacher and student networks contributes to the distillation performance. There are several studies showing two contradictory conclusions; Jin *et al.* [51] and Mirzadeh *et al.* [68] showed that a similar teacher leads to better KD performance; Touvron *et al.* [93] reports that distillation from a substantially different teacher is beneficial for ViT.

We train 25 `ViT-Ti` models with different teacher networks from 69 models that we used by the hard distillation strategy [44]. Experimental details are described in Appendix. Fig. 11a illustrates the relationship between the teacher-student similarity and the distillation performance. Fig. 11a tends to show a not significant negative correlation between teacher-student similarity and distillation performance (-0.32 Pearson correlation coefficient with 0.12 p-value). However, if we only focus on when the teacher and student networks are based on the same architecture (*i.e.*, Transformer), we can observe a strong positive correlation (Fig. 11b) – 0.70 Pearson correlation coefficient with 0.078 p-value. In this case, our observation is aligned with [51, 68]: a teacher similar to the student improves distillation performance. However, when the teacher and student networks are based on different architectures (*e.g.*, CNN), then we can observe a stronger negative correlation (Fig. 11c) with -0.51 Pearson correlation coefficient and 0.030 p-value. In this case, a more dissimilar teacher leads to better distillation performance. We also test other factors that can affect distillation performance in Appendix; We observe that distillation performance is not correlated to teacher accuracy in our experiments.

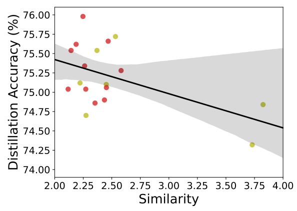
Why do we observe contradictory results for Transformer teachers (Fig. 11b) and other teachers (Fig. 11c)? Here, we conjecture that when the teacher and student networks differ significantly, distillation works as a strong regularizer. In this case, using a more dissimilar teacher can be considered a stronger regularizer



(a) Results on various teachers



(b) Transformer teachers



(c) CNN & other teachers

Fig. 11: Model diversity and distillation performance. (a) We show the relationship between teacher-student similarity and distillation performance of 25 `DeiT-S` models distilled by various teacher networks. We show the relationship when the teacher and student networks are based on (b) Transformer and (c) otherwise.

(Fig. 11c). On the other hand, we conjecture that if two networks are similar, then distillation works as easy-to-follow supervision for the student network. In this case, a more similar teacher will work better because a more similar teacher will provide more easy-to-follow supervision for the student network (Fig. 11b). Our experiments show that the regularization effect improves distillation performance better than easy-to-follow supervision (*i.e.*, the best-performing distillation result is by a CNN teacher). Therefore, in practice, we recommend using a significantly different teacher network for achieving better distillation performance (*e.g.*, using RegNet [74] teacher for ViT student as [93]).

6 Discussion

In Appendix G, we describe more discussions related to SAT. We first propose an efficient approximation of SAT when we have a new model; instead of generating adversarial samples from all models, only generating adversarial samples from the new model can an efficient approximation of SAT (Appendix G.1). We also show that SAT and the same misclassified samples have a positive correlation in Appendix G.2. Appendix G.3 demonstrates that we can estimate the similarity with a not fully trained model (*e.g.*, a model in an early stage). Finally, we describe more possible applications requiring diverse models (Appendix G.4).

7 Conclusion

We have explored similarities between image classification models to investigate what makes the model similar or diverse and whether developing and using diverse models is required. For quantitative and model-agnostic similarity assessment, we have suggested a new similarity function, named SAT, based on attack transferability demonstrating differences in input gradients and decision boundaries. Using SAT, we conduct a large-scale and extensive analysis using 69 state-of-the-art ImageNet models. We have shown that macroscopic architectural properties, such as base architecture and stem architecture, have a more significant impact on similarity than microscopic operations, such as types of convolution, with numerical analysis. Finally, we have provided insight into the ML applications using multiple models based on SAT, *e.g.*, model ensemble or knowledge distillation. Overall, we suggest using SAT to improve methods with multiple models in a practical scenario with a large-scale training dataset and a highly complex architecture.

Acknowledgement

We thank Taekyung Kim and Namuk Park for comments on the self-supervised pre-training. This work was supported by an IITP grant funded by the Korean Government (MSIT) (RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University)) and by the Yonsei Signature Research Cluster Program of 2024 (2024-22-0161).

References

[1] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. In *Adv. Neural Inform. Process. Syst.*, 2021.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Jiawang Bai, Li Yuan, Shu-Tao Xia, Shuicheng Yan, Zhifeng Li, and Wei Liu. Improving vision transformers by revisiting high-frequency components. In *European Conference on Computer Vision*, pages 1–18. Springer, 2022.

[4] Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than cnns? *Advances in neural information processing systems*, 34:26831–26843, 2021.

[5] Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[6] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *Int. Conf. Learn. Represent.*, 2021.

[7] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. In *Int. Conf. Learn. Represent.*, 2021.

[10] Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *Int. Conf. Mach. Learn.*, 2021.

[11] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Int. Conf. Comput. Vis. Worksh.*, 2019.

[12] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Int. Conf. Comput. Vis.*, 2021.

[13] Minghao Chen, Kan Wu, Bolin Ni, Houwen Peng, Bei Liu, Jianlong Fu, Hongyang Chao, and Haibin Ling. Searching the search space of vision transformer. In *Adv. Neural Inform. Process. Syst.*, 2021.

[14] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *Int. Conf. Comput. Vis.*, 2021.

[15] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Adv. Neural Inform. Process. Syst.*, 2017.

[16] Zhengsu Chen, Lingxi Xie, Jianwei Niu, Xuefeng Liu, Longhui Wei, and Qi Tian. Visformer: The vision-friendly transformer. In *Int. Conf. Comput. Vis.*, 2021.

[17] Junsuk Choe, Seong Joon Oh, Sanghyuk Chun, and Hyunjung Akata, Zeynepand Shim. Evaluation for weakly supervised object localization: Protocol, metrics, and datasets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

[18] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[19] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *Adv. Neural Inform. Process. Syst.*, 2021.

[20] Sanghyuk Chun, Seong Joon Oh, Sangdoo Yun, Dongyoon Han, Junsuk Choe, and Youngjoon Yoo. An empirical evaluation on robustness and uncertainty of regularization methods. In *Int. Conf. Mach. Learn. Worksh.*, 2019.

[21] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *Int. Conf. Mach. Learn.*, 2019.

[22] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Int. Conf. Mach. Learn.*, 2020.

[23] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[24] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *Adv. Neural Inform. Process. Syst.*, 2021.

[25] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)*, pages 321–338, 2019.

[26] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Int. Conf. Mach. Learn.*, 2017.

[27] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Int. Conf. Learn. Represent.*, 2021.

[29] Stéphane d'Ascoli, Hugo Touvron, Matthew L Leavitt, Ari S Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. In *Int. Conf. Mach. Learn.*, 2021.

[30] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[31] Yonggan F Fu, Shang Wu, Yingyan Lin, et al. Patch-fool: Are vision transformers always robust against adversarial perturbations? *Int. Conf. Learn. Represent.*, 2022.

[32] Robert Geirhos, Carlos RM Temme, Jonas Rauber, Heiko H Schütt, Matthias Bethge, and Felix A Wichmann. Generalisation in humans and deep neural networks. In *Adv. Neural Inform. Process. Syst.*, 2018.

[33] Robert Geirhos, Kristof Meding, and Felix A Wichmann. Beyond accuracy: quantifying trial-by-trial behaviour of cnns and humans by measuring error consistency. In *Adv. Neural Inform. Process. Syst.*, 2020.

[34] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Int. Conf. Learn. Represent.*, 2015.

[35] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. In *Adv. Neural Inform. Process. Syst.*, 2020.

[36] Chuan Guo, Jared S Frank, and Kilian Q Weinberger. Low frequency adversarial perturbation. *UAI*, 2019.

[37] Dongyoon Han, Sangdoo Yun, Byeongho Heo, and YoungJoon Yoo. Rethinking channel dimensions for efficient model design. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[38] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *Adv. Neural Inform. Process. Syst.*, 2021.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Eur. Conf. Comput. Vis.*, 2016.

[41] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[42] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[43] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. In *Int. Conf. Comput. Vis.*, 2021.

[44] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. In *Adv. Neural Inform. Process. Syst. Worksh.*, 2015.

[45] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[46] Jisung Hwang, Younghoon Kim, Sanghyuk Chun, Jaejun Yoo, Ji-Hoon Kim, and Dongyoon Han. Where to be adversarial perturbations added? investigating and manipulating pixel robustness using input gradients. *ICLR Workshop on Debugging Machine Learning Models*, 2019.

[47] Jaehui Hwang, Jun-Hyuk Kim, Jun-Ho Choi, and Jong-Seok Lee. Just one moment: Structural vulnerability of deep action recognition against one frame attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7668–7676, 2021.

[48] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Adv. Neural Inform. Process. Syst.*, 2019.

[49] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Int. Conf. Mach. Learn.*, 2015.

[50] Mingqi Jiang, Saeed Khorram, and Li Fuxin. Comparing the decision-making mechanisms by transformers and cnns via explanation methods. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9546–9555, 2024.

[51] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In *Int. Conf. Comput. Vis.*, 2019.

[52] Cheng Ju, Aurélien Bibaut, and Mark van der Laan. The relative performance of ensemble methods with deep convolutional neural networks for image classification. *Journal of Applied Statistics*, 45(15):2800–2818, 2018.

[53] Hamid Karimi and Jiliang Tang. Decision boundary of deep neural networks: Challenges and opportunities. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

[54] Gihyun Kim and Jong-Seok Lee. Analyzing adversarial robustness of vision transformers against spatial and spectral attacks. *arXiv preprint arXiv:2208.09602*, 2022.

[55] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Int. Conf. Mach. Learn.*, 2019.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Adv. Neural Inform. Process. Syst.*, 2012.

[57] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207, 2003.

[58] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[59] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Adv. Neural Inform. Process. Syst.*, 2018.

[60] Yuanchun Li, Ziqi Zhang, Bingyan Liu, Ziyue Yang, and Yunxin Liu. Modeldiff: Testing-based dnn similarity comparison for model reuse detection. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 139–151, 2021.

[61] Hanxiao Liu, Andy Brock, Karen Simonyan, and Quoc Le. Evolving normalization-activation layers. In *Adv. Neural Inform. Process. Syst.*, 2020.

[62] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. Pay attention to mlps. In *Adv. Neural Inform. Process. Syst.*, 2021.

[63] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, 2021.

[64] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[65] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Int. Conf. Learn. Represent.*, 2018.

[66] Horia Mania, John Miller, Ludwig Schmidt, Moritz Hardt, and Benjamin Recht. Model similarity mitigates test set overuse. *Adv. Neural Inform. Process. Syst.*, 32, 2019.

[67] Kristof Meding, Luca M Schulze Buschoff, Robert Geirhos, and Felix A Wichmann. Trivial or impossible—dichotomous data difficulty masks model differences (on imagenet and beyond). In *Int. Conf. Learn. Represent.*, 2022.

[68] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

[69] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34:23296–23308, 2021.

[70] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Adv. Neural Inform. Process. Syst.*, 2001.

[71] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[72] Chanwoo Park, Sangdoo Yun, and Sanghyuk Chun. A unified analysis of mixed sample data augmentation: A loss function perspective. In *Adv. Neural Inform. Process. Syst.*, 2022.

[73] Namuk Park and Songkuk Kim. How do vision transformers work? In *Int. Conf. Learn. Represent.*, 2022.

[74] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollar. Designing network design spaces. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[75] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34:12116–12128, 2021.

[76] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[77] Shahbaz Rezaei and Xin Liu. A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning. 2020.

[78] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 2015.

[79] Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which shortcut cues will dnns choose? a study from the parameter-space perspective. In *Int. Conf. Learn. Represent.*, 2022.

[80] Ali Shafahi, Amin Ghiasi, Furong Huang, and Tom Goldstein. Label smoothing and logit squeezing: a replacement for adversarial training? *arXiv preprint arXiv:1910.11585*, 2019.

[81] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Int. Conf. Learn. Represent. Worksh.*, 2014.

[82] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *Int. Conf. Mach. Learn. Worksh.*, 2017.

[83] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[84] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *Int. Conf. Learn. Represent. Worksh.*, 2015.

[85] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[86] Andreas Peter Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. In *Transactions on Machine Learning Research*, 2022.

[87] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Int. Conf. Mach. Learn.*, 2017.

[88] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2014.

[89] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[90] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Int. Conf. Mach. Learn.*, 2019.

[91] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. In *Brit. Mach. Vis. Conf.*, 2019.

[92] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *Adv. Neural Inform. Process. Syst.*, 2021.

[93] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *Int. Conf. Mach. Learn.*, 2021.

[94] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

[95] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv preprint arXiv:1704.03453*, 2017.

[96] Asher Trockman and J. Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

[97] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *Int. Conf. Learn. Represent.*, 2019.

[98] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inform. Process. Syst.*, 2017.

[99] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[100] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2020.

[101] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(10):3349–3364, 2020.

[102] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks.

In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[103] Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1924–1933, 2021.

[104] Futa Waseda, Sosuke Nishikawa, Trung-Nghia Le, Huy H Nguyen, and Isao Echizen. Closer look at the transferability of adversarial examples: How they fool different models differently. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023.

[105] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[106] Yuxin Wu and Kaiming He. Group normalization. In *Eur. Conf. Comput. Vis.*, 2018.

[107] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[108] I Zeki Yalniz, Hervé Jégou, Kan Chen, Manohar Paluri, and Dhruv Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv preprint arXiv:1905.00546*, 2019.

[109] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[110] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022.

[111] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Int. Conf. Comput. Vis.*, 2019.

[112] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Brit. Mach. Vis. Conf.*, 2016.

[113] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conf. Learn. Represent.*, 2018.

[114] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2022.

[115] Linjun Zhang, Zhun Deng, Kenji Kawaguchi, Amirata Ghorbani, and James Zou. How does mixup help with robustness and generalization? In *Int. Conf. Learn. Represent.*, 2021.

[116] Qilong Zhang, Xiaodan Li, Yuefeng Chen, Jingkuan Song, Lianli Gao, Yuan He, and Hui Xue. Beyond imagenet attack: Towards crafting adversarial examples for black-box domains. *arXiv preprint arXiv:2201.11528*, 2022.

[117] Richard Zhang. Making convolutional networks shift-invariant again. In *Int. Conf. Mach. Learn.*, 2019.

[118] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Ö Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.